

IN THE CLAIMS

Please amend the claims as follows.

- 1 1. (Previously Presented) An apparatus comprising:
 - 2 at least one processor;
 - 3 a memory coupled to the at least one processor;
 - 4 a first compilation unit residing in the memory, the first compilation unit
 - 5 comprising a plurality of object oriented classes that are part of an object oriented
 - 6 program, wherein the object oriented program is defined by the combination of the first
 - 7 compilation unit and at least one other compilation unit; and
 - 8 a compiler residing in the memory and executed by the at least one processor in a
 - 9 partial compilation environment, the compiler allocating at least one object in the first
 - 10 compilation unit to an invocation stack frame for a method in the first compilation unit
 - 11 that allocates the at least one object, wherein the compiler comprises:
 - 12 an escape analysis mechanism that operates on the first compilation unit
 - 13 prior to a second compilation unit and that marks each instruction in the first
 - 14 compilation unit that allocates a new object as one of global escape, no escape,
 - 15 and arg escape based on information available from classes visible in the first
 - 16 compilation unit but not visible in the uncompiled second compilation unit; and
 - 17 an object allocation mechanism that allocates at least one object that is
 - 18 created by an instruction marked as no escape by the escape analysis mechanism
 - 19 to an invocation stack frame for a method that allocates the object.

- 1 2. (Cancelled)

1 3. (Previously presented) The apparatus of claim 1 wherein the escape analysis
2 mechanism marks each instruction in the first compilation unit that allocates a new object
3 as one of global escape, no escape, and arg escape based on information available from
4 classes visible in the first compilation unit and from classes that are outside the first
5 compilation unit that are visible in a specified classpath.

1 4. (Previously Presented) An apparatus comprising:
2 at least one processor;
3 a memory coupled to the at least one processor;
4 a first compilation unit residing in the memory, the first compilation unit
5 comprising a plurality of object oriented classes that are part of an object oriented
6 program, wherein the object oriented program is defined by the combination of the first
7 compilation unit and at least one other compilation unit; and
8 a compiler residing in the memory and executed by the at least one processor in a
9 partial compilation environment, the compiler allocating at least one object in the first
10 compilation unit to an invocation stack frame for a method in the first compilation unit
11 that allocates the at least one object
12 wherein the compiler comprises:
13 a code generator that creates two versions of code for a selected object method, a
14 first version using stack allocation of objects and a second version using heap allocation
15 of objects; and
16 a run time code selector that selects one of the first and second versions to execute
17 at run time based on a determination of whether classes seen at run time match expected
18 classes within predetermined limits.

1 5. (Previously Presented) An apparatus comprising:
2 at least one processor;
3 a memory coupled to the at least one processor;
4 a first compilation unit residing in the memory, the first compilation unit
5 comprising a plurality of object oriented classes that are part of an object oriented
6 program, wherein the object oriented program is defined by the combination of the first
7 compilation unit and at least one other compilation unit; and
8 a compiler residing in the memory and executed by the at least one processor in a
9 partial compilation environment, the compiler comprising:
10 an escape analysis mechanism that operates on the first compilation unit
11 prior to a second compilation unit and that marks each instruction in the first
12 compilation unit that allocates a new object as one of global escape, no escape,
13 and arg escape based on information available from classes visible in the first
14 compilation unit but not visible in the uncompiled second compilation unit and
15 from classes that are outside the first compilation unit that are visible in a
16 specified classpath;
17 an object allocation mechanism that allocates at least one object that is
18 created by an instruction marked as no escape by the escape analysis mechanism
19 to an invocation stack frame for a method that allocates the object;
20 a code generator that creates two versions of code for a selected object
21 method, a first version using stack allocation of objects and a second version
22 using heap allocation of objects; and
23 a run time code selector that selects one of the first and second versions to
24 execute at run time based on a determination of whether classes seen at run time
25 match expected classes within predetermined limits.

1 6. (Previously Presented) A method for allocating objects to memory in an object
2 oriented program that comprises a first compilation unit and a second compilation unit,
3 the method comprising the steps of:
4 (A) compiling the first compilation unit;
5 (B) during the compiling of the first compilation unit and before the compilation
6 of the second compilation unit, marking each instruction that allocates a new object as
7 one of global escape, no escape, and arg escape based on information available from
8 classes in the first compilation unit and from classes that are outside the first compilation
9 unit that are visible in a specified classpath; and
10 allocating at least one object that is created by an instruction marked as no escape
11 by the escape analysis mechanism to an invocation stack frame for a method that allocates
12 the at least one object.

1 7. (Cancelled)

1 8. (Original) The method of claim 6 wherein step (B) comprises the steps of:
2 creating two versions of code for a selected object method, a first version using
3 stack allocation of objects and a second version using heap allocation of objects; and
4 selecting at run time one of the first and second versions to execute at run time
5 based on a determination of whether classes seen at run time match expected classes
6 within predetermined limits.

1 9. (Original) In an object oriented computer program that comprises a first compilation
2 unit and at least one other compilation unit, a method for allocating objects in the first
3 compilation unit to memory, the method comprising the steps of:
4 marking each instruction that allocates a new object as one of global escape, no
5 escape, and arg escape based on information available from classes in the first
6 compilation unit and from classes that are outside the first compilation unit that are
7 visible in a specified classpath;
8 creating two versions of code for a selected object method, a first version using
9 stack allocation of objects and a second version using heap allocation of objects; and
10 selecting at run time one of the first and second versions to execute at run time
11 based on a determination of whether classes seen at run time match expected classes
12 within predetermined limits.

1 10. (Currently amended) A program product comprising:
2 a compiler that compiles in a partial compilation environment a first compilation
3 unit comprising a plurality of object oriented classes that are part of an object oriented
4 program, wherein the object oriented program is defined by the combination of the first
5 compilation unit and at least one other compilation unit, the compiler allocating at least
6 one object in the first compilation unit to an invocation stack frame for a method in the
7 first compilation unit that allocates the at least one object;
8 wherein the compiler comprises:
9 an escape analysis mechanism that operates on the first compilation unit
10 prior to a second compilation unit and that marks each instruction in the first
11 compilation unit that allocates a new object as one of global escape, no escape,
12 and arg escape based on information available from classes visible in the first
13 compilation unit; and
14 an object allocation mechanism that allocates at least one object that is
15 created by an instruction marked as no escape by the escape analysis mechanism
16 to an invocation stack frame for a method that allocates the object;
17 wherein the escape analysis mechanism marks each instruction in the first
18 compilation unit that allocates a new object as one of global escape, no escape,
19 and arg escape based on information available from classes visible in the first
20 compilation unit but not visible in the uncompiled second compilation unit and from
21 classes that are outside the first compilation unit that are visible in a specified
22 classpath; and
23 recordable signal bearing media bearing the compiler.

1 11. (Cancelled)

1 12. (Cancelled)

1 13. (Cancelled)

1 14. (Cancelled)

1 15. (Original) The program product of claim 10 wherein the compiler comprises:

2 a code generator that creates two versions of code for a selected object method, a
3 first version using stack allocation of objects and a second version using heap allocation
4 of objects; and

5 a run time code selector that selects one of the first and second versions to execute
6 at run time based on a determination of whether classes seen at run time match expected
7 classes within predetermined limits.

1 16. (Currently amended) A program product comprising:
2 (A) a compiler that compiles a first compilation unit comprising a plurality of
3 object oriented classes that are part of an object oriented program, wherein the object
4 oriented program is defined by the combination of the first compilation unit and at least
5 one other compilation unit, the compiler comprising:
6 (A1) an escape analysis mechanism that marks each instruction that
7 allocates a new object as one of global escape, no escape, and arg escape based on
8 information available from classes in the first compilation unit and from classes
9 that are outside the first compilation unit that are visible in a specified classpath;
10 (A2) an object allocation mechanism that allocates at least one object that
11 is created by an instruction marked as no escape by the escape analysis
12 mechanism to an invocation stack frame for a method that allocates the object;
13 (A3) a code generator that creates two versions of code for a selected
14 object method, a first version using stack allocation of objects and a second
15 version using heap allocation of objects; and
16 (A4) a run time code selector that selects one of the first and second
17 versions to execute at run time based on a determination of whether classes seen
18 at run time match expected classes within predetermined limits; and
19 (B) recordable signal bearing media bearing the compiler.

1 17. (Cancelled)

1 18. (Cancelled)